Iterative Span Selection: Self-Emergence of Resolving Orders in Semantic Role Labeling

Shuhei Kurita[1,2] **Hiroki Ouchi**[3,1] **Kentaro Inui**[4,1] **Satoshi Sekine**[1]
[1]RIKEN [2]JST PRESTO [3]Nara Institute of Science and Technology [4]Tohoku University
shuhei.kurita@riken.jp hiroki.ouchi@is.naist.jp
inui@tohoku.ac.jp satoshi.sekine@riken.jp

Abstract

Semantic Role Labeling (SRL) is the task of labeling semantic arguments for marked semantic predicates. Semantic arguments and their predicates are related in various distinct manners, of which certain semantic arguments are a necessity while others serve as an auxiliary to their predicates. To consider such roles and relations of the arguments in the labeling order, we introduce iterative argument identification (IAI), which combines global decoding and iterative identification for the semantic arguments. In experiments, we first realize that the model with random argument labeling orders outperforms other heuristic orders such as the conventional left-to-right labeling order. Combined with simple reinforcement learning, the proposed model spontaneously learns the optimized labeling orders that are different from existing heuristic orders. The proposed model with the IAI algorithm achieves competitive or outperforming results from the existing models in the standard benchmark datasets of span-based SRL: CoNLL-2005 and CoNLL-2012.

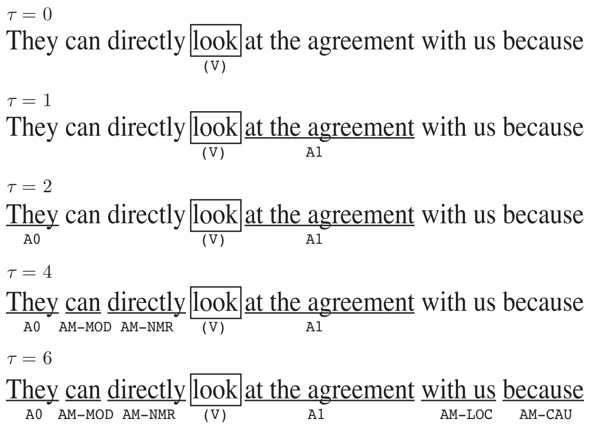
$\tau = 0$
They can directly look at the agreement with us because
(V)
$\tau = 1$
They can directly look at the agreement with us because
(V) A1
$\tau = 2$
They can directly look at the agreement with us because
A0 (V) A1
$\tau = 4$
They can directly look at the agreement with us because
A0 AM-MOD AM-NMR (V) A1
$\tau = 6$
They can directly look at the agreement with us because
A0 AM-MOD AM-NMR (V) A1 AM-LOC AM-CAU

Figure 1: Example of the iterative argument identification with the given predicate "look". At the bottom of the final state $\tau = 6$, all semantic arguments are presented with spans and labels. The time step τ corresponds the iterative process of the proposed model.

1 Introduction

Semantic role labeling (Carreras and Màrquez, 2004, 2005) is the task of identifying and resolving the relations between semantic predicates and their arguments based on the PropBank (Kingsbury and Palmer, 2002) predicate-argument structure. In span-based SRL, semantic predicates comprise several semantic arguments that are expressed as spans of tokens in the sentence. Recent span-based SRL models incorporate neural networks into a global decoding approach. Here syntactic features are injected into the neural network model (Strubell et al., 2018), span-based scoring for semantic argument is adapted (Ouchi et al., 2018), and the unified representations for both span-based and dependency-based SRL are applied (Li et al., 2019; Zhou et al., 2020). However, in these approaches, the labeling order is not determined inside the neural network and hence models require some extrapolated graph decoding procedures, similar to the graph-based approaches that rely on external graph decoding (Lewis et al., 2015). Such external decoding procedures are typically not trained during the model training and hence hinder accurate decoding.

The sequential labeling approach is another major branch of span-based SRL models (Màrquez et al., 2005; Zhou and Xu, 2015; He et al., 2017; Tan et al., 2018; Li et al., 2020), wherein the models resolve sentences from the beginning to the end or *left-to-right* ordering by attaching labels that represent both semantic spans and roles. Li et al. (2020) proposed the BIO labeling-based model with predefined regularizers of unique case roles, exclusively overlapping roles and PropBank frame definitions. However, sequential labeling approaches often suffer from the error-propagation problem(Senge et al., 2014; Dinarelli and Tellier, 2018). One reason is that they are not able to arrange the argument identification orderings in decoding.

Thus, in this study, we explore an SRL model that combines global decoding and iterative labeling approaches: iterative argument identification (IAI). Our model works iteratively: the model identifies one argument individually and stores it for each time step. The stored semantic arguments are used as “clues” for identifying other arguments in later time steps. Moreover, our models can identify semantic arguments from arbitrary orders because our model identifies arguments from any part of the sentence. This means that our model can consider relations of predicates and arguments in the decoding order. In SRL, semantic arguments have various roles to their predicates. Figure 1 represents an example of attached semantic role labels for the partial sentence of “They can directly look at the agreement with us because...”. In this example, both arguments “They” and “at the agreement” represent crucial semantic roles to their predicate “look.” However, other arguments such as “with us” and the phrase following “because” represent additional information to their predicate.

As identified arguments become clues in later processes, choosing suitable decoding orderings affects the final performance of the proposed model because many clues become available to identify a new argument in later time steps. Here we ask the following question: Are there certain labeling orderings that can identify arguments more accurately than heuristic orderings? Empirical experiments revealed that the traditional *left-to-right* ordering although strong, is not the best ordering, e.g., simple *random* ordering in imitation learning outperform the *left-to-right* ordering. Based on the results obtained, we explored models that follow better decoding orders than the heuristic orders. We assume optimal transition paths are not generated with heuristics or hand-engineering, and rather expect the self-emergence of the optimal transition paths that are different from the existing heuristic transition paths through the model training. We applied simple policy-gradient-based reinforcement-learning for the IAI model and found that reinforcement learning slightly leverages the model performance thereby allowing models to arrange orderings resulting in different argument orderings from existing heuristics, which was confirmed through several analyses. Furthermore, our model achieved competitive or better performances than the existing models in the standard benchmark datasets.[1]

[1]The code is available at `https://github.com/shuheikurita/iss_srl`

2 Related Work

The idea of optimizing the labeling orders in decoding is a branch of the *easy-first* strategy (Tsuruoka and Tsujii, 2005; Goldberg and Elhadad, 2010; Ma et al., 2013; Martins and Kreutzer, 2017). In SRL, Wolfe et al. (2016) proposed the SRL model with the pseudo teacher approaches for the processing orders in SRL. They exploit violation fixing perceptron and their parser explores the states of the highest scored path along with the word frequency ordering baseline. Since their proposed model of “easy-first dynamic” follows the highest scoring action, their model explores limited transition spaces during training. Refinement of existing SRL is also examined in dependency-based SRL (Lyu et al., 2019; Chen et al., 2019). Reinforcement learning is also applied in broad syntactic and semantic parsing studies (Lê and Fokkens, 2017; Fried and Klein, 2018; Naseem et al., 2019; Kurita and Søgaard, 2019). Multi-task neural network is often applied to such structured syntactic analyses (Søgaard and Goldberg, 2016; Kurita et al., 2017). It is notable that adversarial training is also applied to extract knowledge from unannotated corpora in Japanese predicate-argument structure analysis (Kurita et al., 2018).

Lattice-based approach is also a promising approach for SRL in traditional (Täckström et al., 2015) and neural models (FitzGerald et al., 2015). However, they rely on external dynamic programming decoding. Choi and Palmer (2011) proposed the transition-based model for dependency-based SRL. They applied a set of transition actions that are similar to the shift-reduce parser (Nivre, 2008) in syntactic parsing. They also adapted the self-learning clustering technique for predicates that are unseen in training. Blloshmi et al. (2021) address a sequence-to-sequence labeling model which performs competitive with sequence-labeling models. Indeed, most of the recent SRL resolving studies address the global-decoding approach (Ouchi et al., 2018; Li et al., 2018, 2019; Zhou et al., 2020; Conia and Navigli, 2020) or the sequential labeling approach (Shi and Lin, 2019; Li et al., 2020; Marcheggiani and Titov, 2020; Zhang et al., 2021; Kasai et al., 2019) in both span-based and dependency-based SRL. In this paper, we introduce the iterative approach for the global argument selection and enable models to determine the ordering of resolving semantic arguments with modern neural networks and reinforcement learning for span-based SRL.

3 Model

3.1 Iterative argument identification

The span-based SRL model predicts multiple spans of tokens as semantic arguments for each marked semantic predicate and attaches semantic role labels to the arguments. Some semantic arguments have crucial roles in the grammatical or semantic structures of a sentence, whereas other arguments have rather auxiliary roles to their predicates. Such arguments are, therefore, more difficult to resolve than others. In iterative span selection, our model repeatedly predicts one semantic argument for each semantic predicate in a single iteration. The previously predicted semantic arguments are stored in a partial semantic arguments buffer. In later time steps, our model is able to use the information from the previously extracted semantic arguments to effectively predict the remaining arguments.

For a given predicate p, the proposed model determines the next argument boundary and its role label in each iteration. Formally, let X is the input sentence, p is a marked predicate and $\mathbf{Y}_p^g$ be the set of all annotated arguments of the marked predicate p in the annotated data g. One semantic argument $y_{i,p} \in \mathbf{Y}_p^g$ is represented by the span of tokens and its semantic role label as $y_{i,p} = \{t^s, t^e, l\}$. Here, t^s is the beginning of the argument span, t^e is the end of the span, and l is the attached semantic role label. Then, we define a transition action $\mathbf{a}_p$ for each predicate p. The action $\mathbf{a}_p$ includes the decision of whether the predicate p has more unresolved arguments or not, and the detection of a new single semantic argument $y_{i,p} = \{t^s, t^e, l\}$. In each iteration, the model resolve a new semantic role label of $y_{i,p} = \{t^s, t^e, l\}$ by choosing t^s, t^e and l respectively, or decide that the predicate p has no more semantic arguments. When the model predicts semantic arguments for the marked predicates, the resolved arguments are stored in the partial SRL buffer of that predicate. The partial SRL buffer contains the previously predicted arguments $\mathbf{Y}_p^\tau$ for each predicate p in the iteration of the time step τ. The partial SRL buffer is updated after each transition and used as part of the model input in the next step. Note that the transitions are independently performed for each predicate. Therefore, the model can stop transitions for some predicates while the model continues transitions for other predicates. The structure of the partial semantic argument buffer is explained in Section 3.2.2.

3.2 Neural network

Our neural network model predicts the probabilities of the transition action $\mathbf{a}_p$ as $p(\mathbf{a}_p|X, p, \mathbf{Y}_p^\tau)$ for all predicates in each iteration τ. Figure 2 represents the neural network model. The network consists of three parts: (i) the sentence encoder, (ii) the partial SRL encoder, and (iii) the span selection and labeling decoder.

3.2.1 Sentence encoder

For the sentence encoder, we use the self-attention architecture of transformer (Vaswani et al., 2017), which is compatible with the huge pretrained language encoder models, such as BERT (Devlin et al., 2019). Pretrained models often rely on sub-word segmentations while SRL is a token-level task. For a token with multiple sub-tokens, we use the beginning sub-token for the entire representation of the original token. We initially split the input sentence into sub-tokens and add the special tokens of "`[NULL]`", "`[EOS]`" and "`[PAD]`". Here "`[NULL]`" has the special meaning that the predicate has no unresolved arguments. Following the pretrained models, we apply wordpiece (Wu et al., 2016) for the original tokens to obtain sub-words. The phrase of "the amended filings", for example, becomes the sequence of sub-tokens as "`[NULL]` the amended filing #s `[EOS] [PAD] ... [PAD]`" where the token "filings" are split into two sub-tokens "filing" and "#s".

We apply transformer to encode a sequence of sub-tokens in the sentence to obtain $h(t_i) \in R^d$ for the representation of the i-th sub-token t_i. d is the output dimension of the transformer model. In contrast to the representations of the partial semantic argument buffer, the obtained representations $h(t_i)$ for the sentence are not altered during transitions.

3.2.2 Partial SRL encoder

We employ a special encoder that directly encodes the partially-extracted semantic arguments of $\mathbf{Y}_p^\tau$ that are resolved in the former transitions of the iterative argument identification algorithm. We present the partial SRL buffers $\mathbf{Y}_p^\tau$ which contain the spans of previously extracted semantic arguments in Figure 3. We prepare the same number of the partial SRL buffers with the number of the marked predicates in the sentence. During SRL resolving, partial SRL buffers are updated separately for each predicate. Contents of the partial SRL buffer for a predicate does not affect the arguments identification for other predicates. This nature al-

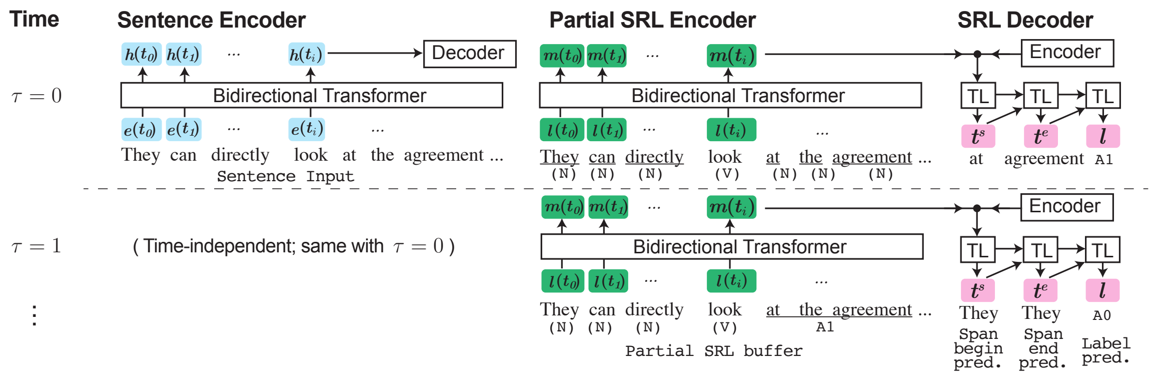
Time
Sentence Encoder
Partial SRL Encoder
SRL Decoder
$\tau = 0$
$h(t_0)$ $h(t_1)$ ... $h(t_i)$
Decoder
Bidirectional Transformer
$e(t_0)$ $e(t_1)$ ... $e(t_i)$ ...
They can directly look at the agreement ...
Sentence Input
$m(t_0)$ $m(t_1)$ ... $m(t_i)$
$l(t_0)$ $l(t_1)$ ... $l(t_i)$ ...
(N) (N) (N) (V) (N) (N) (N)
Encoder
TL
t^s t^e l
at agreement A1
$\tau = 1$
(Time-independent; same with $\tau = 0$)
A1
Partial SRL buffer
They They A0
Span begin pred.
Span end pred.
Label pred.

Figure 2: Overall network architecture: the sentence encoder, partial SRL encoder, and SRL decoder. The sentence encoder takes inputs of the (sub)token representation of $e(t_i)$ and computes the time-independent sentence representation of $h(t_i)$, for each sub-token t. The partial SRL encoder takes inputs of the label representation of $l(t_i)$ and computes the time-dependent partial SRL buffer representation of $m(t_i)$. In the partial SRL buffer, (V) represents the marked predicates and (N) represents the token does not have the attached labels yet. We depict the time sequence $\tau = 0$ and $\tau = 1$ cases for the same example with Figure 1.

Time	They	can	directly	look	at	the	agreement
$\tau = 0$	(N)	(N)	(N)	(V)	(N)	(N)	(N)
$\tau = 1$	(N)	(N)	(N)	(V)	A1	A1	A1
$\tau = 2$	A0	(N)	(N)	(V)	A1	A1	A1
$\tau = 3$	A0	(N)	AM-MNR	(V)	A1	A1	A1

Figure 3: Example of the partial SRL buffer updates for the predicate “look” of the phrase “They can directly look at the agreement”.

Name	Sent. Enc.	SRL Enc.	SRL Dec.
Hidden size	1024	256	2560
Transformer layers	12	3	3 (Total)

Table 1: Hyper-parameters for transformers of sentence encoder, SRL encoder and the SRL decoder. The other hyperparameters are the same with those of BERT.

lows the parallelization of SRL resolving for each predicate as discussed in the Appendix A.6.

At the initial transition of $\tau = 0$, a partial SRL buffer is filled with “(N)” labels for all sub-tokens except the marked semantic predicate with a label of “(V)”. Here “(N)” label represents that the token has no predicted role labels yet for the predicate marked “(V)”. At the end of τ-th transition, the model updates the SRL buffers with new predicted semantic arguments that are used for the next transition of $\tau + 1$.

We use a transformer-based encoder for the sequence of extracted SRL spans to obtain the partial semantic argument representations of $m^\tau(t_i) \in \mathcal{R}^d$ of the i-th sub-token t_i. This transformer is different from the sentence encoder transformer and it is not pretrained. As the partial SRL buffers are updated in transitions, the representations for them are altered. Therefore the partial SRL representations contribute to changes in the SRL resolving actions.

3.2.3 SRL decoder

We employ a decoder network for incrementally predicting the beginning and end of the new argument span for each predicate in the sentence. Argument prediction is performed by predicting the beginning token t^s, the span end token t^e and the argument label l. To do so, the decoder network predicts the probabilities of the next action $\mathbf{a}_p$ for each predicate p with the inputs of the sentence, the predicate and partial semantic role label buffer: $p(\mathbf{a}_p|\mathrm{X}, p, \mathbf{Y}_p^\tau)$. The action $\mathbf{a}_p$ consists of three decisions: (i) choosing the beginning of the span t^s or deciding this predicate does not have further semantic arguments e.g., the model selects the “[NULL]” token as t^s, (ii) choosing the end of the span t^e and (iii) attaching a semantic role label l for the predicted span $[t^s, t^e]$ of the argument.

The decoder network works as follows. First, the model concatenates the representations of sub-tokens $\mathbf{h}$ and the partial SRL buffer m^τ for τ-th transition and input it into transformer layers for the scoring tokens as the beginning of the span $s_s(\cdot)$ with a softmax function over sub-tokens

$$p(t^s = t_i) = \frac{\exp\left(s_s([h(t_i), m^\tau(t_i)])\right)}{\sum_{t'} \exp\left(s_s([h(t'), m^\tau(t')])\right)} \quad (1)$$

to obtain the probability $p(t^s)$ for a sub-token t_i to become the beginning of the span t^s. Sub-tokens that are not the beginning of the original tokens don’t become t^s. Therefore the probability $p(t^s)$ is re-normalized for these beginning sub-tokens while the probabilities of other sub-tokens for t^s are adjusted to 0. In the evaluation, we choose the beginning of the next argument span with $\arg\max_t p(t^s)$. In imitation learning, we choose the teacher label of the beginning of the span t^s_g from the annotated arguments. In reinforcement learning, we choose the teacher labels with Gumbel-Softmax here. In the sampling and evaluation, models are required to consider sub-tokens that are the beginning of some original tokens as the candidates of the next argument beginning t^s.

Similarly, the model predicts the end of the span t^e given the sub-token of the span beginning t^s. Similar to the beginning of the span, We prepare transformer layers for $s_e(\cdot)$ with a softmax function over sub-tokens

$$p(t^e = t_i) = \frac{\exp\left(s_e([h(t_i), m^\tau(t_i), h(t^s)])\right)}{\sum_{t'} \exp\left(s_e([h(t'), m^\tau(t'), h(t^s)])\right)}$$

to obtain the probability $p(t^e)$ for a sub-token t_i as the end of the argument span t^e. Here the model uses representations of the sub-tokens of the argument beginning t^s that are resolved first. To reduce the size of the concatenated vectors of the three representations, we extract the first half values of the three vectors and concatenate them. The model does the sampling from $p(t^e)$ with Gumbel-Softmax to obtain t^e similar to t^s during reinforcement learning. Finally, the model computes the label distribution of the argument from the sentence and the SRL representation $h(t)$ and $m(t)$ as the beginning and end tokens representation of the argument span, $[h(t^s), m(t^s)]$ and $[h(t^e), m(t^e)]$ with a scoring function $s_l(\cdot)$ of another transformer layer:

$$p(l) = \frac{\exp\left(s_l([h(t^s), m^\tau(t^s), h(t^e), m^\tau(t^e)])\right)}{\sum_l \exp\left(s_l([h(t^s), m^\tau(t^s), h(t^e), m^\tau(t^e)])\right)}$$

for the label l prediction.

3.3 Learning

The problem in training IAI is that there are no annotated orders for determining SRLs. In Figure 3, we present an example of transitions for IAI. However, such teacher orders are not always available during training models. For training models, there are two possible approaches: *imitation learning* as the teacher path is given by an oracle and *reinforcement learning* as the model explores the transition paths during training.

3.3.1 Imitation learning

We first define teacher transition paths. Given all annotated arguments $\mathbf{Y}^g_p$ for each predicate, the transition path is the sequence of the annotated arguments $\{y_0, \cdots, y_T\}$. We prepare simple heuristic transition paths: *right-to-left*, *left-to-right*, *close-to-distant*, *distant-to-close* and *random* orders. For each semantic predicate, the left-to-right order teacher selects the annotated semantic arguments from left to right. This is similar behaviour to the transition-based models. The right-to-left order teacher is the inverse of the left-to-right. The close-to-distant and distant-to-close order teachers select arguments based on the distance of sub-tokens from the predicate.[2] These four teacher transition paths always yield the same transition paths, whereas the random transition teacher yields different transition paths in each epoch. Therefore, the random transition benefits from this de-facto data augmentation.[3] We compare the results of those heuristic teachers in imitation learning in Appendix A.4.

3.3.2 Reinforcement learning

In reinforcement learning, the model determines the transition path during training. In particular, we apply a policy gradient to explore the transition space that is uncommon during the imitation training. A Gumbel-Softmax distribution (Jang et al., 2017) has the essential property that it can be smoothly annealed into a categorical distribution. Thus we use Gumbel-Softmax for the sampling from the next possible transitions.

3.3.3 Rewards for reinforcement learning

We exploit simple immediate rewards for reinforcement learning. We apply the positive reward of $r = 1$ for all transitions of the correct arguments and the negative reward of $r = -1$ for all incorrect transitions. In each transition, the model determines the beginning of the next span t^s, the ending of the span t^e and the label l incrementally. If the model identifies one of the correct arguments from the remaining unresolved arguments, it gets the $r = 3$ positive rewards in total in a single transition. When the model makes a wrong prediction, it receives the $r = -1$ negative reward at this time

[2] If two arguments are at the same distance in the number of sub-tokens from the predicate, we regard the left argument as close to the predicate for the convenience.

[3] The reinforcement learning can also benefits from this de-facto data augmentation. However, it would be less effective than those for *random* because of the limited transition paths.

Model	CoNLL-2005									CoNLL-2012					
	Dev			WSJ			Brown			Dev			Test		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Ouchi+2018 ELMo	87.4	86.3	86.9	88.2	87.0	87.6	79.9	77.5	78.7	87.2	85.5	86.3	87.1	85.3	86.2
Ouchi+2018 ELMo (E)	88.0	86.9	87.4	**89.2**	87.9	88.5	81.0	78.4	79.6	**88.6**	85.7	87.1	**88.5**	85.5	87.0
Li+ 2019 ELMo	-	-	-	85.2	87.5	86.3	74.7	78.1	76.4	-	-	-	84.9	81.4	83.1
Shi+2019 BERT	-	-	-	88.6	89.0	88.8	81.9	82.1	82.0	-	-	-	85.9	87.0	86.5
Zhou+2020 BERT	-	-	-	89.04	88.79	88.91	81.89	80.98	81.43	-	-	-	-	-	-
Li+2020 BERT*	-	-	-	-	-	-	-	-	-	85.97	86.38	86.18	85.82	86.36	86.09
Li+2020 RoBERTa	87.24	87.26	87.25	88.05	88.00	88.03	80.04	79.56	79.80	86.60	86.89	86.74	86.40	86.83	86.61
Zhang+2021 BERT	-	-	-	87.54	88.32	87.93	81.91	82.37	82.14	-	-	-	85.93	87.32	86.62
Left-to-right	87.70	88.16	87.93	88.76	88.94	88.85	82.40	82.59	82.50	87.28	87.83	87.55	87.35	87.89	87.62
Random	87.86	88.30	88.08	88.73	89.07	88.90	82.69	**83.33**	83.01	87.57	87.68	87.62	87.59	87.76	87.67
Random+RL	**88.32**	**88.23**	**88.28**	89.18	**89.11**	**89.15**†	**83.63**	83.13	**83.37**†	87.94	**87.39**	**87.67**	88.04	**87.50**	**87.77**

Table 2: The empirical results in CoNLL-2005 and CoNLL-2012 datasets in labeled attachment score (LAS). Li+2020 uses the original BERT finetuned twice, marked as $*$. (E) denotes the result of the model ensemble. Bold fonts for the best results. We present the averaged result of the three runs with different seeds for `Random+RL`. We confirmed the statistical significance of the test set results in bootstrapped paired t-test in $p < 0.05$ denoted as $\dagger$.

and it cannot obtain further rewards in this transition. Even if the model made wrong predictions for some arguments in the past transitions, the model is still allowed to obtain positive rewards when the model identifies other correct arguments in later transitions. For example, if a model makes a correct prediction of t_s and an incorrect prediction of t_e for an argument, the model obtains the $r = 1$ reward for the t_s prediction and the $r = -1$ reward for the t_e prediction. This model cannot obtain any rewards regardless of the label l prediction for this argument. However, this model is still allowed to obtain further rewards when it predicts other arguments in later transitions. When the model selects the special token `NULL` which represents the stop iteration, the model obtains the reward of $r = 1$ if the model has resolved all the correct arguments; otherwise, $r = 0$.

We summarize further reinforcement learning details and implementation details in Appendix A.1.

4 Experiments

We conducted experiments with the datasets provided from CoNLL-2005 and 2012 shared tasks (Carreras and Màrquez, 2005; Pradhan et al., 2012). We followed the standard splits of the datasets provided from CoNLL and used the official and standard evaluation script.[4] In CoNLL-2005, sections 2nd-21st of the Wall Street Journal (WSJ) corpus are for the training set and section 24th for the development set. The WSJ section 23rd is for the

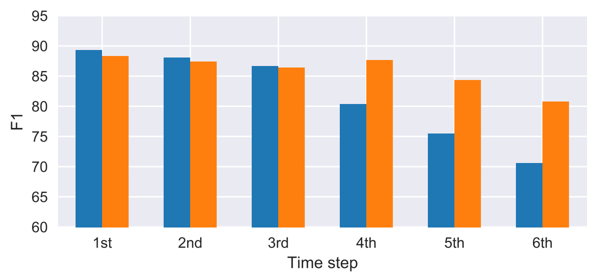

Figure 4: The transition step-wise F1 score in LAS for the first to sixth transition on the development set of CoNLL-2005. Navy (left) for `Random` model and orange (right) for `Random+RL` model.

in-domain test set while Brown corpus 3rd sections is for the out-of-domain test set. CoNLL-2012 dataset is from the OntoNotes v5.0 corpus. See Pradhan et al. (2013) for more details of CoNLL-2012. In the evaluation, the SRL labels of “V” are omitted following the official evaluation script because they are obvious from the marked predicates.

4.1 Comparison with previous results

First, we compared our models of the iterative argument identification algorithm to the previous state-of-the-art models, including the global decoding model (Zhou et al., 2020) and the variants of sequence labeling-based model (Shi and Lin, 2019; Li et al., 2020; Zhang et al., 2021). These models use the pretrained models of BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). We additionally included the two graph-based model of Ouchi et al. (2018) and Li et al. (2019) with ELMo (Peters et al., 2018) for reference. Note that

[4]This script is available at: `https://www.cs.upc.edu/~srlconll/soft.html`

Label	A0	A1	A2	A3	AM-ADV	AM-DIS	AM-LOC	AM-MNR	AM-MOD	AM-NEG	AM-TMP	R-A0	R-A1
F1													
Random	93.93	90.25	82.96	81.40	67.90	80.11	70.69	70.74	98.07	94.14	88.38	96.10	92.26
Random+RL	94.03	90.46	82.79	83.16	69.12	79.39	72.22	70.88	98.27	93.66	88.28	96.83	93.07
Δ	0.10	0.21	-0.17	1.76	1.22	-0.72	1.53	0.14	0.20	-0.48	-0.10	0.73	0.81
Avg. Steps													
Random	1.96	1.63	2.19	2.52	2.73	2.67	2.26	2.25	2.36	2.66	2.65	2.01	1.42
Random+RL	2.70	1.31	1.59	1.90	2.37	2.58	2.60	1.86	2.57	2.03	2.70	2.50	1.62
Δ	0.74	-0.32	-0.60	-0.62	-0.36	-0.09	0.34	-0.39	0.21	-0.63	0.05	0.49	0.20

Table 3: **Top**: label-wise performance by F1 score. **Bottom**: average step times when argument labels are accurately attached. Results in the development set of CoNLL-2005. Δ is the difference between `Random` and `Random+RL` models. Here, we present 13 label types that appear most frequently in the dataset.

Zhou et al. (2020) also uses syntactic information in the multi-task training and hence the results are not directly comparable.[5]

Table 2 shows the performance of the two heuristic order models of `Left-to-right` and `Random` and the proposed `Random+RL` model that determines the optimal parsing path during training. We also compared results of these models with the performance of previous models. Our model achieves better results than the model of Zhou et al. (2020) that relies on the pretrained model of BERT-large and syntactic information. Li et al. (2020) also use the special BERT-large model that is finetuned twice by the authors. Among all models, the proposed `Random+RL` model achieves the best performance in the F1 scores in both the development set and test set of the CoNLL-05 and CoNLL-12 datasets. We also confirm that our `Random+RL` outperforms other heuristics such as the model trained in `Left-to-right` manner in F1 score as discussed in Appendix A.4.

Paolini et al. (2021) proposed a pre-trained T5-base model (Raffel et al., 2020) for multiple tasks. We noticed that TANL with a single dataset is comparable with our experimental setting even though the pretrained model is quite different. Although they achieved 89.3 in F1 score of WSJ of CoNLL-05, our model out-performs their model performance of 82.0 in F1 in Brown of CoNLL-05. Our `Random+RL` model achieves competitive performance with their 87.7 in F1 of CoNLL-12.

4.2 Does Reinforcement learning help?

We apply reinforcement learning (RL) for the model trained with the random ordering. In Table 2, we confirm that reinforcement learning slightly improves the performance in both CoNLL-05 and CoNLL-12. We further investigate the reasons of this performance gain and notice that reinforcement learning surely changes the argument identifications in the later time steps of transitions. Figure 4 presents the comparisons of LAS scores for the predicted arguments in each transition step. In first to third transitions, there are no large differences between the `Random` and `Random+RL` models. However, the `Random+RL` model retains the performance in the later transitions. Although this result is contrary to the intuition of the existing "*easy-first*" strategy, we assume this is one of the reasons why reinforcement learning enhances the final model performance.

We take a close look at the effect of reinforcement learning on the label prediction accuracy. Table 3 presents the two detailed results: the performance comparison and the average transition steps required to identify arguments for each label type. We firstly notice that the `Random+RL` model achieve better or competitive accuracy except of A2, AM-DIS and AM-NEG. We also assume that the `Random+RL` model follows some specific orders in identifying arguments. For some labels such as A3 and AM-ADV, the model chooses to label them first. For some arguments, such as A0 and AM-LOC, the model chooses to label them later.

4.3 How does RL affect SRL ordering?

Here we provide further analyses for the relation of the semantic roles and the identification ordering of labels. The semantic roles of arguments come from

[5]In Zhou et al. (2020), they also reported scores with XLNet. However, we cannot reproduce their XLNet results in any efforts. They didn't release their codes for XLNet and even didn't reply our emails for asking training details. We therefore decide not to include their XLNet results in Table 2.

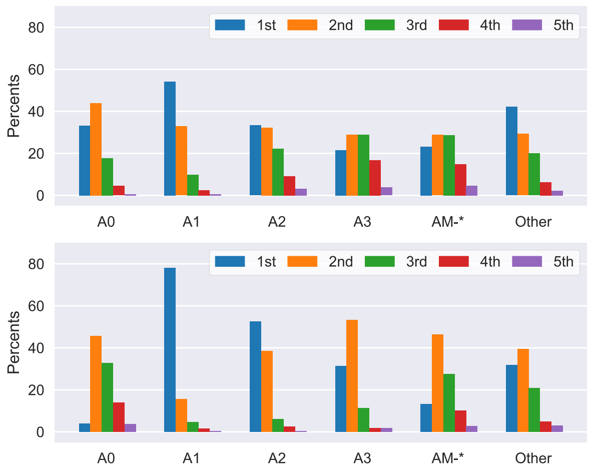

Figure 5: The ratio (%) of the label types identified in from 1st to 5th transitions on the development set of CoNLL-2005. **Top**: imitation learning (`Random`). **Bottom**: reinforcement learning (`Random+RL`).

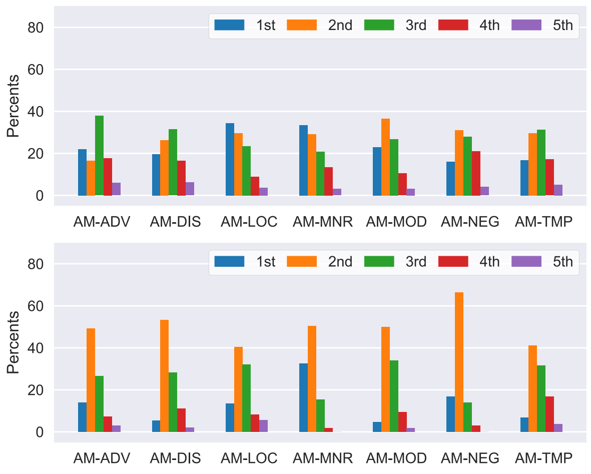

Figure 6: The ratio (%) of the label types of the modifiers (AM-*) identified in from 1st to 5th transitions on the development set of CoNLL-2005. **Top**: imitation learning (`Random`). **Bottom**: reinforcement learning (`Random+RL`).

PropBank frames annnotation guidelines.[6] A0 and A1 labels correspond to external and internal arguments in the government and binding theory. They are either subject or object roles depending on the transitive and intransitive verbs. AM-* arguments are modifiers and other labels include referential expressions. We analyze how the resolving orders are affected by these label roles as a result of reinforcement learning. In Figure 5, we present the ratio of the resolved transition steps for each label class. For example, the sharp peak of 1st transition for A1 with `Random+RL` at the bottom of Figure 5 means nearly 80% of A1 labels in the development set are identified in the first transition.

Here we notice the `Random+RL` model has a clear tendency of resolving A1 or A2 first if they exist. It also seems that the `Random+RL` model prefers to identify A0, A3, or AM-* arguments in 2nd or 3rd transitions. We assume this corresponds to the importance of such role labels for predicates. Although the `Random` model has a similar tendency in some labels, it is less obvious.

Figure 6 presents the details of modifier labels. We present the seventh most frequent modifier argument roles in SRL here. TMP, LOC, DIS, NEG, MOD, ADV and MNR labels correspond to temporal, locative, discourse markers, negation, modals, adverbials, and manner markers respectively. We cannot read strong preferences of resolving orders for the `Random` model. There are several inconsistent orders: AM-LOC in `Random` has a weak peak at the 1st transition, while AM-TMP has a weak peak at the 3rd transition. The `Random+RL` model consistently identifies these arguments mostly in the 2nd transition if they exist. It is also interesting that manner markers have the preference to be identified first and the negation has a strong peak at 2nd transition. Other AM-* modifiers are mostly processed in the 2nd or 3rd transitions by the `Random+RL` model.

As seen in Figure 4, reinforcement learning improves the labeling accuracy, especially in the later transition steps. For later transition steps, the model uses previously resolved arguments as "clues" to identify the remaining arguments. Therefore the model tunes which argument to identify first and later via reinforcement learning as the `Random+RL` model introduce specific orders in labeling in Figure 5 and Figure 6. As a result, the model retains the labeling performance in later transitions and hence it outperforms the existing heuristic approaches such as the left-to-right order and random ordering in SRL.

5 Conclusion

We develop the iterative argument identification (IAI) algorithm for the global decoding and iterative resolving for span-based SRL. Our model with IAI is capable of identifying semantic arguments one by one in arbitrary orders. In empirical experiments, we enhance our model with policy-

[6] `https://verbs.colorado.edu/~mpalmer/projects/ace/PBguidelines.pdf`

gradient-based transition exploration. Our model out-performs the existing models with the same pre-trained model in both CoNLL-05 and CoNLL-12 datasets. In the analyses, we confirm that reinforcement learning enable models to learn a different resolving orders from existing heuristic orders and slightly enhance the performance, which suggest the emergence of the transition path through the training.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 22K17983 and 19K2035, and by JST PRESTO Grant Number JPMJPR20C2.

References

Rexhina Blloshmi, Simone Conia, Rocco Tripodi, and Roberto Navigli. 2021. Generating senses and roles: An end-to-end model for dependency- and span-based semantic role labeling. In *IJCAI*.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 89–97, Boston, Massachusetts, USA. Association for Computational Linguistics.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan. Association for Computational Linguistics.

Xinchi Chen, Chunchuan Lyu, and Ivan Titov. 2019. Capturing argument interaction in semantic role labeling with capsule networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5415–5425, Hong Kong, China. Association for Computational Linguistics.

Jinho D. Choi and Martha Palmer. 2011. Transition-based semantic role labeling using predicate argument clustering. In *Proceedings of the ACL 2011 Workshop on Relational Models of Semantics*, pages 37–45, Portland, Oregon, USA. Association for Computational Linguistics.

Simone Conia and Roberto Navigli. 2020. Bridging the gap in multilingual semantic role labeling: a language-agnostic approach. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1396–1410, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Marco Dinarelli and Isabelle Tellier. 2018. New recurrent neural network variants for sequence labeling. In *Computational Linguistics and Intelligent Text Processing*, pages 155–173, Cham. Springer International Publishing.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970, Lisbon, Portugal. Association for Computational Linguistics.

Daniel Fried and Dan Klein. 2018. Policy gradient as a proxy for dynamic oracles in constituency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 469–476, Melbourne, Australia. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: NAACL*, pages 742–750, Los Angeles, California.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. 2017. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*.

Jungo Kasai, Dan Friedman, Robert Frank, Dragomir Radev, and Owen Rambow. 2019. Syntax-aware neural semantic role labeling with supertags. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 701–709, Minneapolis, Minnesota. Association for Computational Linguistics.

Paul Kingsbury and Martha Palmer. 2002. From TreeBank to PropBank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA).

Shuhei Kurita, Daisuke Kawahara, and Sadao Kurohashi. 2017. Neural joint model for transition-based Chinese syntactic analysis. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1204–1214, Vancouver, Canada. Association for Computational Linguistics.

Shuhei Kurita, Daisuke Kawahara, and Sadao Kurohashi. 2018. Neural adversarial training for semi-supervised Japanese predicate-argument structure analysis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 474–484, Melbourne, Australia. Association for Computational Linguistics.

Shuhei Kurita and Anders Søgaard. 2019. Multi-task semantic dependency parsing with policy gradient for learning easy-first strategies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2420–2430, Florence, Italy. Association for Computational Linguistics.

Minh Lê and Antske Fokkens. 2017. Tackling error propagation through reinforcement learning: A case of greedy dependency parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 677–687, Valencia, Spain. Association for Computational Linguistics.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG parsing and semantic role labelling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1444–1454, Lisbon, Portugal. Association for Computational Linguistics.

Tao Li, Parth Anand Jawale, Martha Palmer, and Vivek Srikumar. 2020. Structured tuning for semantic role labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8402–8412, Online. Association for Computational Linguistics.

Zuchao Li, Shexia He, Jiaxun Cai, Zhuosheng Zhang, Hai Zhao, Gongshen Liu, Linlin Li, and Luo Si. 2018. A unified syntax-aware framework for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2401–2411, Brussels, Belgium. Association for Computational Linguistics.

Zuchao Li, Shexia He, Zhao Hai, Yiqing Zhang, Zhuosheng Zhang, Xi Zhou, and Xiaodong Zhou. 2019. Dependency or span, end-to-end uniform semantic role labeling. In *AAAI*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. Cite arxiv:1907.11692.

Chunchuan Lyu, Shay B. Cohen, and Ivan Titov. 2019. Semantic role labeling with iterative structure refinement. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1071–1082, Hong Kong, China. Association for Computational Linguistics.

Ji Ma, Jingbo Zhu, Tong Xiao, and Nan Yang. 2013. Easy-first POS tagging and dependency parsing with beam search. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–114, Sofia, Bulgaria. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2020. Graph convolutions over constituent trees for syntax-aware semantic role labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3915–3928, Online. Association for Computational Linguistics.

Lluís Màrquez, Pere Comas, Jesús Giménez, and Neus Català. 2005. Semantic role labeling as sequential tagging. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 193–196, Ann Arbor, Michigan. Association for Computational Linguistics.

André F. T. Martins and Julia Kreutzer. 2017. Learning what's easy: Fully differentiable neural easy-first taggers. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 349–362, Copenhagen, Denmark. Association for Computational Linguistics.

Tahira Naseem, Abhishek Shah, Hui Wan, Radu Florian, Salim Roukos, and Miguel Ballesteros. 2019. Rewarding Smatch: Transition-based AMR parsing with reinforcement learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4586–4592, Florence, Italy. Association for Computational Linguistics.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Hiroki Ouchi, Hiroyuki Shindo, and Yuji Matsumoto. 2018. A span selection model for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1630–1642, Brussels, Belgium. Association for Computational Linguistics.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *9th International Conference on Learning Representations, ICLR 2021*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Robin Senge, Juan José del Coz, and Eyke Hüllermeier. 2014. On the problem of error propagation in classifier chains for multi-label classification. In *Data Analysis, Machine Learning and Knowledge Discovery*, pages 163–170, Cham. Springer International Publishing.

Peng Shi and Jimmy J. Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *ArXiv*, abs/1904.05255.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, Berlin, Germany. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium. Association for Computational Linguistics.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep semantic role labeling with self-attention. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4929–4936. AAAI Press.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 467–474, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. pages 5–32. Springer.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Travis Wolfe, Mark Dredze, and Benjamin Van Durme. 2016. A study of imitation learning methods for semantic role labeling. In *Proceedings of the Workshop on Structured Prediction for NLP*, pages 44–53, Austin, TX. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Zhisong Zhang, Emma Strubell, and Eduard H. Hovy. 2021. Comparing span extraction methods for semantic role labeling. In *SPNLP*.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137, Beijing, China. Association for Computational Linguistics.

Junru Zhou, Zuchao Li, and Hai Zhao. 2020. Parsing all: Syntax and semantics, dependencies and spans. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4438–4449, Online. Association for Computational Linguistics.

A Appendix

A.1 Details for reinforcement learning

We apply the cross-entropy loss for imitation learning and the policy gradient (Williams, 1992) for reinforcement learning. For reinforcement learning, we firstly train models with imitation learning until the learning converges, and then finetune it with the policy gradient. We select the best performance of the models at the development set in both imitation learning and reinforcement learning. The reinforcement learning for the argument span is conducted as follows. First, we compute a probability of each (sub-)token becoming the *beginning* of the next argument. We apply sampling over this probability and determine the beginning (sub-)token of the next argument span. With this sampled beginning (sub-)token, our model similarly compute another probability of each (sub-)token becoming the *end* of the next argument span. Again we apply sampling over the probability and determine the end (sub-)token of the next argument span. Finally, the model attaches the label to the sampled span.

We apply Gumbel-softmax for sampling of the next argument to resolve from possible transition paths during the training. For Gumbel-softmax, the inverse temperature parameter β becomes a hyper-parameter. If β is too large, the model samples from very limited transition paths that are close to the narrow path of $\arg\max(\pi_i)$. Thus it gets stuck in the local optima. If β is too small, the model samples from various transition paths that include unrealistic arguments and hence hinder convergence of the training. We perform experiments of training SRL models with $\beta \in \{0.1, 0.5, 1, 2, 3\}$ and report the best performance result of $\beta = 0.5$ for CoNLL-05 and $\beta = 0.1$ for CoNLL-12.

A.2 Training Details

In terms of the batch size, we notice that the larger batch size helps the training. We conducted experiments with the batch size of $[16, 32, 64, 128]$ and obtained the best result at 128. We use the transformer implementation of Hugging Face (Wolf et al., 2020). We apply the pretrained BERT-Large model of Bert-large-cased-whole-word-masking for the sentence encoder, and therefore the hyper-parameters of the sentence encoder are the same as those of the pretrained BERT-Large model with capitalized tokens and whole-word masking. For the partial SRL encoder and the SRL decoder models, we use the hyper-parameters in Table 1. We train our model on machines with four NVIDIA V100 GPU cards. We obtained similar results only with a single NVIDIA V100 GPU card combined with the gradient accumulation.

A.3 What order does the reinforcement learning model prefer to follow?

We further investigate in what orders the models prefer to identify arguments. Here, we analyze which role label the model tends to identify first for a pair of arguments that have the same predicate. Figure 7 represents the heat-map for visualizing the ratio for pairs of semantic role labels before and after each transition. Given the number $N_{X,Y}$ of pairs of arguments for the role label X (in the horizontal axis) and Y (in the vertical axis), we count the cases that role label X is processed *after* the role label Y as $n_{X,Y}$, and we plot $n_{X,Y}/N_{X,Y}$.

In the `Random+RL`, we easily notice that there is a consistent tendency that the model identifies the A0 labels later than any other labels. We check the transition paths of the model processing outputs and confirm that the model frequently identifies A0 labels at last. We also notice that, in Figure 5, A0 has the higher ratio for 3rd, 4th, and 5th transitions than others in `Random+RL`. This might be related to the position of A0 in syntactic trees. We also confirm that the model chooses the A1, A2, and A3 labels first and the AM-* labels later. Among the AM-* labels, AM-NEG and AM-NEG are frequently processed first, although they are mostly processed later than A1 and A2. Overall, the proposed `Random+RL` model identifies semantic role labels and spans as follows: A1 and A2 first, AM-* and other labels later, and A0 label at last. The `Random` model doesn't have such obvious tendencies at a glance.

A.4 Is the traditional left-to-right resolving good for the IAI algorithm?

Contrary to the intuition, we notice that the traditional left-to-right ordering doesn't achieve the best performance for the IAI algorithm among the heuristic orderings. We train our models with five different teacher orders: right-to-left, left-to-right, close-to-distant, distant-to-close, and random as Sec. 3.3.1. The results are shown in Table 4. We notice that the model with the random order teacher performs best in both the in-domain test set of WSJ and the out-of-domain test set of Brown Corpus in CoNLL-05. Similar tendency has observed in the CoNLL-12 dataset. These experiments remind us

Model	CoNLL-2005									CoNLL-2012					
	Dev			WSJ			Brown			Dev			Test		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Left-to-right	87.70	88.16	87.93	88.76	88.94	88.85	82.40	82.59	82.50	87.28	87.83	87.55	87.35	87.89	87.62
Right-to-left	87.98	88.39	88.18	88.76	88.91	88.83	82.31	82.19	82.25	87.34	**88.00**	87.66	87.27	**87.94**	87.60
Close-to-dist.	87.69	**88.44**	88.06	88.50	88.92	88.71	81.95	82.78	82.36	87.25	87.90	87.57	87.18	87.90	87.53
Dist.-to-close	87.76	88.07	87.91	88.68	89.03	88.85	82.33	82.50	82.41	87.10	87.73	87.41	87.04	87.70	87.37
Random	87.86	88.30	88.08	88.73	89.07	88.90	82.69	**83.33**	83.01	87.57	87.68	87.62	87.59	87.76	87.67
Random+RL	**88.32**	88.23	**88.28**	**89.18**	**89.11**	**89.15**	**83.63**	83.13	**83.37**	**87.94**	87.39	**87.67**	**88.04**	87.50	**87.77**

Table 4: The empirical results in CoNLL-2005 and CoNLL-2012 datasets in LAS. We compare five model with different teacher orders in the training: `Left-to-right`, `Right-to-left`, `Close-to-dist.`, `Dist.-to-close` and `Random` and with reinforcement learning (`Random+RL`). "Dev" is the result in the development set. Bold fonts for the best results. We present the averaged scores of the three runs with different seeds.

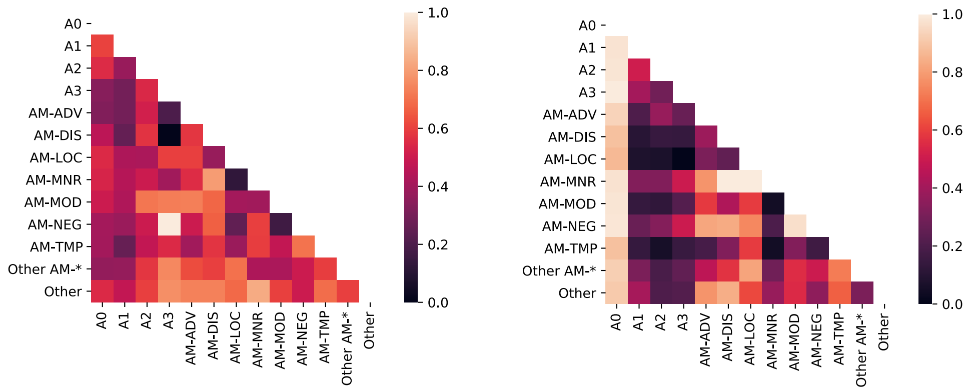

Figure 7: The ratio for the role labels, on the horizontal axis, that are identified *after* the role labels on the vertical axis. The bright color represents the labels on the horizontal axis is likely to be identified *after* the labels on the vertical axis. **Left**: imitation learning (`Random`). **Right**: reinforcement learning (`Random+RL`). Analysis on the development set of CoNLL-2005.

that adapting traditional heuristic ordering is not the best way to train the IAI models. We explore orderings that are better than these heuristics.

A.5 How reinforcement learning affects the argument distance from the predicates?

Figure 8 presents the distribution of the distance from predicates to their argument. We draw four distribution lines that correspond to the 1st, 2nd, 3rd and 4th transitions. Here we count the number of sub-tokens between the predicates and their arguments as the distance. Arguments at the right of their predicates have the positive distance and other arguments have the negative distance. In both `Random` and `Random+RL`, models tend to choose the arguments that are placed right after the predicates. However, this tendency becomes clear in `Random+RL`: the model firstly chooses the arguments right after the predicates and later this model chooses arguments that are placed before the predicates. This suggests that the model learns the new ordering of the argument identification during reinforcement learning.

A.6 Computation times and speed analysis

Iterative argument identifications requires $O(PA)$-times transitions for a sentence that has P predicates and the maximum number of arguments A in theory. However, iterative argument identification has two properties that make it possible to speed up and parallelize the computation. First, the pretrained transformer-based sentence representations are unchanged during the parsing. This reduce the computation cost. Second, the transitions in iterative argument identifications are independently performed for each predicate. Therefore we can paralleize the transitions for each predicate on the same minibatch of the neural network. Thanks to this predicate-parallelization, the computation times for the overall neural network become the number of argument A if they are on the same minibatch. The average processing speed is 7.5 sentences per second when the minibatch size for evaluation is 48 on a single GPU of NVIDIA V100.

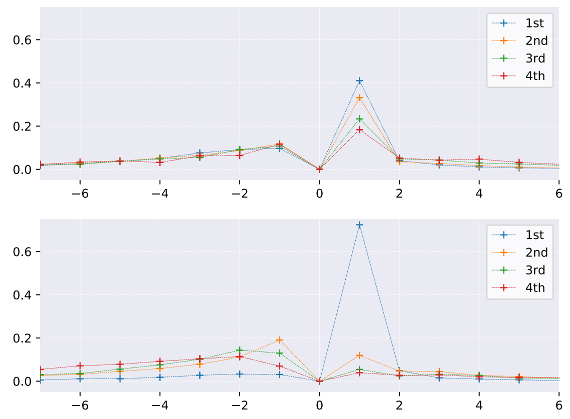

Figure 8: The relation of the argument resolving orders and the distance of predicates and arguments. We represent the first four transitions. **Top**: imitation learning (Random). **Bottom**: reinforcement learning (Random+RL).

A.7 Limitations and potential risks

This work addresses the tools that are developed with the dataset and pretrained models that are widely shared in our community. If the original datasets or pretrained models contain potential risks, our tool might be affected by them. We will take careful looks to prevent our tools from potential abuses.